1

Appearance-Based Vision and the Automatic Generation of Object Recognition Programs

Keith D. Gremban and Katsushi Ikeuchi

1 July 1992
CMU-CS-92-159

School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, Pennsylvania 15213-3890

This report will appear in the collection *3D Object Recognition Systems*, edited by Anil K. Jain and Patrick J. Flynn, to be published by Elsevier.

This research was sponsored by the Avionics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U.S. Air Force, Wright-Patterson AFB, Ohio 45433-6543 under Contract F33615-90-C-1465, ARPA Order No. 7597. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or of the U.S. Government.

Keywords: computer vision, vision and scene understanding, automatic programming

Abstract

The generation of recognition programs by hand is a time-consuming, labor-intensive task that typically results in a special purpose program for the recognition of a single object or a small set of objects. Recent work in automatic code generation has demonstrated the feasibility of automatically generating object recognition programs from CAD-based descriptions of objects. Many of the programs which perform automatic code generation employ a common paradigm of utilizing explicit object and sensor models to predict object appearances; we refer to the paradigm as appearance-based vision, and refer to the programs as vision algorithm compilers (VACs). A CAD-like object model augmented with sensor-specific information like color and reflectance, in conjunction with a sensor model, provides all the information needed to predict the appearance of an object under any specified set of viewing conditions. Appearances, characterized in terms of feature values, can be predicted in two ways: analytically, or synthetically. In relatively simple domains, feature values can be analytically determined from model information. However, in complex domains, the analytic prediction method is impractical. An alternative method for appearance prediction is to use an appearance simulator to generate synthetic images of objects which can then be processed to extract feature values. In this paper, we discuss the paradigm of appearance-based vision and present in detail two specific VACs: one that computes feature values analytically, and a second that utilizes an appearance simulator to synthesize sample images.

1 Introduction

The generation of object recognition programs by hand is a time-consuming, labor-intensive process that typically results in a special purpose program for the recognition of a single object or a small set of objects. The reason for this lies in the design methodology: on the basis of a representative set of sample images, the designer selects a set of image features and specifies the procedure to be used in matching image features to object features. Feature selection requires application of many different operators to the sample images in order to determine the best feature set. Optimization of both feature extraction operators and the matching procedure requires "tweaking" of parameters through extensive experimentation. The entire process requires a highly skilled and motivated designer. While the result is often an efficient, robust solution to the problem, the overall cost is so high as to be prohibitive for many applications.

Recently, driven in part by an effort to make computer vision systems more economically feasible, research has been conducted towards the goal of automatically generating recognition code from a CAD-based description of an object. Most current industrial parts are designed and manufactured using computer-aided tools, so CAD descriptions exist for most parts; automatic generation of recognition code from the same model information used for design and manufacture would be an efficient, cost-effective approach. A number of programs for automatic generation of object recognition code have been written, and many of these programs employ a common paradigm in which explicit object and sensor models are used to predict object appearances. We refer to the paradigm as *appearance-based vision*, and programs which generate object recognition programs are called *vision algorithm compilers*, or VACs.

Appearance-based vision represents an extension to the familiar paradigm of model-based vision. Model-based vision defines an execution-time methodology of matching observed image features to model features, but does not address the issue of defining a methodology for selecting either features or matching procedures. Appearance-based vision systems employ model-based matching during the execution-time recognition phase, but also employ a characteristic methodology during the off-line compilation phase, during which features are selected and processing strategies determined.

In principle, a CAD-like object model, augmented with sensor-specific information like surface color, roughness, and reflectance, can be used in conjunction with a sensor model to predict the appearance of the object under any specified set of viewing conditions. For example, knowing the color and reflectance of a polygonal patch permits a complete determination of its variation in appearance with respect to a video camera and a fixed light source. VACs can predict object appearances in two different ways: analytically, or synthetically.

In relatively simple domains, feature values can be analytically determined from model information. For example, the set of visible edges of a polyhedron with lambertian surfaces is a straightforward computation. Similarly, the collection of visible surfaces with respect to a range sensor can be easily computed. As objects and their properties grow in complexity, however, effects such as self-shadowing and inter-reflection become more important, but are difficult to incorporate into an analysis. Analytic prediction of appearances is therefore impractical for some domains.

An alternative to analytic prediction of appearances is the use of an appearance simulator. An appearance simulator generates synthetic images of objects under specific viewing conditions, with respect to a given sensor. An appearance simulator can be used to generate a representative collection of sample images, which can then be processed and analyzed to extract the feature values that characterize object appearances. Thus, a VAC utilizing an appearance simulator is a computational implementation of the traditional hand-generation approach to building object recognition systems.

In this paper, we discuss the paradigm of appearance-based vision. In the next section, we review the state-of-the-art in appearance-based vision and the automatic generation of object recognition programs. In the course of the review, the defining characteristics of appearance-based vision systems will be noted. Following the review, we will present in detail two VACs which typify appearance-based systems. In section 3, we present a VAC that employs analytic prediction of appearances, and the advantages and limitations of this approach are discussed. Then, in section 4, we present a VAC that utilizes an appearance simulator. A brief summary concludes the paper.

2 The Paradigm of Appearance-Based Vision

The history of computer vision research has largely been a study in making vision systems work. Little attention has been paid to the study of how to design and build application systems. For example, the dominant paradigm in computer vision is that of model-based vision. Briefly, the model-based paradigm can be characterized as *hypothesize-predict-verify*: given a collection of image features, *hypothesize* a match of an image feature to a model feature; use the hypothesized match to *predict* the image locations of other model features; *verify* the predictions and update the hypothesis. The paradigm does not specify how to select the features to use, or how to perform the matching. The paradigm defines an approach to execution-time processing, rather than an approach to system building.

Typically, a model-based vision system is built by hand through a time-consuming, labor-intensive, experimental process. The designer specifies a task scenario and obtains sample images. The images are repeatedly processed using a variety of operators until the designer has selected a set of features that is adequate for the task. The designer then determines appropriate values to characterize the features, implements a matching procedure, and experimentally "tunes" the procedure to perform optimally. The resulting programs are often extremely efficient and competent for the intended application, but are difficult to extend or modify. Moreover, every model-based system takes about the same amount of effort to develop.

Because every computer vision system is essentially a custom solution to a specific problem, a typical computer vision system is expensive to develop and install, and is capable of recognizing only a single part or a small number of parts under very special conditions. Modifications to existing systems are difficult to make, and the cost to develop a new system is as high as that of the first system. Clearly this is an unacceptable situation. For computer vision systems to be practical, they must be cost-effective. This means, among other things, that a computer vision system must be economical to develop, install, and modify.

Appearance-based vision addresses the problem of building cost-effective computer vision systems; it specifies a methodology for the automatic generation of object recognition programs. Appearance-based vision is an extension of the model-based paradigm that formalizes and automates the design process. Appearance-based vision can be characterized as an automated process of analyzing the appearances of objects under specified observation conditions, followed by the automatic generation of model-based object recognition programs based on the preceding analysis. An appearance-based system is called a vision algorithm compiler, or VAC.

The appearance of an object is a function of both the properties of the object, and the properties of the sensor system as well. Object models must be more sophisticated than conventional CAD models, which only represent object geometry. Models for appearance-based vision must include any information that contributes to the appearance of an object with respect to a sensor. For example, 3D geometry is necessary to predict apparent shape, but must be augmented with information about surface roughness, reflectance, transmittance, and color. Since the appearance of an object varies with respect to the sensor, sensor models must also be specified. A sensor model must include information about the relative geometry of the illuminant and detectors, as well as information about the features detectable by the sensor.

One characteristic of appearance-based vision is that both objects and sensors are explicitly modeled, and therefore exchangeable. Hence, a given VAC can generate object recognition code for many different objects using the same sensor model, or the set of objects can be fixed and the sensor models varied. Appearance-based systems embody an abstract principle of appearance prediction based on the use of accurate models.

The task of object recognition is composed of two subtasks: *object identification*, in which objects in the scene are identified, and *object localization*, in which the exact pose (position and orientation) of an identified object is computed. A given object recognition task may consist of either or both subtasks, and VACs have been constructed to solve all the different combinations of identification and localization.

A VAC incorporates a two stage approach to object recognition. The first stage is executed off-line and consists of performing analysis of predicted object appearances and the generation of object recognition code. The second stage is executed on-line, and consists of applying the previously generated code to input images. The first stage is executed only once for a given object recognition task, and can be relatively expensive. The second stage is executed many times, and must be both fast and cost-effective. The high cost of the first stage is amortized over a large number of executions of the second stage.

During the first, off-line stage of processing, the appearances of the object are predicted over the expected range of viewpoints. The predicted appearances are analyzed to determine the set of features that are useful for recognition. Frequently, identification and localization are performed most efficiently using different feature sets. Once the feature set is determined, representative values for the features are determined and compiled into a recognition strategy.

The second, on-line stage of an appearance-based system is nothing more than the run-time execution of the generated strategy. Since there are many different computational strategies, the on-line stage varies considerably between systems. The important principle is that extensive off-line analysis can be used to make the on-line stage as efficient, robust, and cost-effective as possible.

In the next subsection, we present an historical overview of research on appearance-based vision. Then, building on the historical perspective, we enumerate and elaborate on the commonalities between the systems; it is this set of common characteristics that define the paradigm of appearance-based vision.

2.1 Historical Perspective

Goad [9] presented an early version of a VAC. He noted that the computational activity of an application vision system could be split up into two stages: an analysis stage, in which useful information about the task can be compiled; and an execution stage in which the compiled information is utilized to perform object recognition. Moreover, the compilation stage is performed off-line only once, at considerable computational expense, while the execution stage is executed on-line many times, and should be optimized to be as rapid as possible. The computational expense of the off-line stage is then offset by the savings realized by the repeated execution of the optimized on-line stage.

In Goad's system, an object is described by a list of edges and a set of visibility conditions for each edge. Visibility is determined by checking visibility at a representative number of viewpoints obtained by tessellating the viewing sphere. Object recognition is performed by a process of iteratively matching object and image edges until either a satisfactory match is found, or the algorithm fails. The sequence of matchings is compiled during the off-line analysis phase. Goad's system was not completely automatic, however. Goad selected edges as the features to be used for recognition, and the order of edge matching was specified by hand.

The 3DPO system of Bolles and Horaud [4] was built with the intended goal of using off-line analysis to produce the fastest, most efficient on-line object recognition program possible. 3DPO utilized the local-feature-focus method, in which a prominent *focus* feature is initially identified, and then secondary features predicted from the focus feature are used to fine-tune the localization result. The system was not fully automatic, as the focus features and secondary features were chosen by hand.

Ikeuchi and Kanade [14], [16] first pointed out the importance of modeling sensors as well as objects in order to predict appearances, and noted that the features that are useful for recognition depend on the sensor being used. Their system, which will be elaborated on in section 3 predicts object appearances at a representative set of viewpoints obtained by tessellating the viewing sphere. The appearances are grouped into equivalence classes with respect to the visible features; the equivalence classes are called *aspects*. A recognition strategy is generated from the aspects and their predicted feature values, and is represented as an interpretation tree. Each interpretation tree specifies the sequence of operations required to precisely localize an object. The sequence of operations is broken up into two parts: the first part classifies an input image into an instance of one of the aspects, while the second part determines the precise pose (position and orientation) of the object within the specified aspect.

Hansen and Henderson [10] demonstrated a system that analyzed 3D geometric properties of objects and generated a recognition strategy. The system was developed to make use of a range sensor for recognition. The system examines object appearances at a representative set of viewpoints obtained by tessellating the viewing sphere. Geometric features at each viewpoint are examined, and the properties of robustness, completeness, consistency, cost, and uniqueness are evaluated in order to select a complete and consistent set of features. For each model, a strategy tree is constructed, which describes the search strategy used to recognize and localize objects in a scene. Each strategy tree first uses the strongest set of features to identify the object and aspect, uses secondary features to corroborate the aspect identification, and then to find the exact pose.

The system of Arman and Aggarwal [1] was designed to be capable of selecting the proper sensor for a given task. Starting with a CAD model of an object, the system builds up a tree in which the root node represents the object, and the leaves represent features (where features are dependent upon the sensor selected), and a path from the root to a leaf passes through nodes representing increasing specificity. For example, starting at the root, a path could lead to a node representing the sensor property (shape, color, reflectance,...), then to a feature class node (surface, boundary,...), and so on down to a leaf node that represents a particular feature of the object. Each arc in the tree is weighted by a "reward potential" that represents the likely gain from traversing that link. At run time, the system traverses the tree from the root to the leaves, choosing the branch with the highest weight at each level, and backtracking when necessary.

The PREMIO system of Camps, et al [5] predicts object appearances under various conditions of lighting, viewpoint, sensor, and image processing operators. Unlike other systems, PREMIO also evaluates the utility of each feature by analyzing the detectability, reliability, and accuracy. The predictions are then used by a probabilistic matching algorithm that performs the on-line process of identification and localization.

The BONSAI system of Flynn and Jain [7] identifies and localizes 3D objects in range images by comparing relational graphs extracted from CAD models to relational graphs constructed from range image segmentation. The system constructs the relational graphs off-line using two techniques: first, view-independent features are calculated directly from a CAD model; second, synthetic images are constructed for a representative set of viewpoints obtained by tessellating the viewing sphere, and the predicted areas of patches are determined and stored as an attribute of the appropriate relational graph node. During the on-line recognition phase, an interpretation tree is constructed which represents all possible matchings of the graph constructed from a range image, and the stored model graph. Recognition is performed by heuristic search of the interpretation tree.

Sato, et al [20] demonstrated a system for recognition of specular objects. This system will be discussed more completely in section 4. During an off-line phase, the system generates synthetic images from a representative set of viewpoints. Specularities are extracted from each image, and the images are grouped into aspects according to shared specularities, and each specularity is evaluated in terms of its detectability and reliability. A deformable template is also prepared for each aspect. At execution time, an input image is classified into a few possible aspects using a continuous classification procedure based on Dempster-Shafer theory. Final verification and localization is performed using deformable template matching.

2.2 The Common Threads

After reviewing the different appearance-based systems that have been constructed, it is useful to go back and point out the common processing steps.

2.2.1 Two Phases of Processing

Each of the systems discussed employ two distinct phases of processing. The first phase, variously called *off-line*, *compilation*, or *analysis*, consists of analyzing object appearances and constructing recognition strategies. The second phase, called *on-line*, *run-time*, or *execution*, consists of applying the strategies generated in the first phase to an actual recognition task. In general, computational efficiency is not a concern in the first phase, since it does not directly affect the actual time or effort required to perform object recognition, and the cost is only incurred once. In contrast, the second phase is expected to execute many times as part of an application, and consequently must be efficient. In effect, the time spent during strategy generation can be amortized over the number of executions of the resultant strategy.

2.2.2 Explicit Object and Sensor Models

Any object recognition system must match the appearance of an object with respect to some sensor, to a model of the object. Consequently, to automatically generate a program for object recognition, it is necessary to predict and analyze object appearances. Objects appear differently to different sensors, so in order to predict object appearances, sensors must be modeled as well. An appearance-based system therefore includes both object and sensor models.

The early appearance-based systems only made use of explicit object models and utilized implicit sensor models, although the need for different types of models to represent different types of detectable features was acknowledged. All recent systems have emphasized the fact that appearance depends upon the sensor and include explicit models of both objects and sensors. Models may be exchanged so that the same VAC can generate object recognition programs for a variety of objects and sensors.

2.2.3 Appearance Prediction and Analysis

In general, there are two approaches to predicting object appearances: analytic and synthetic. The analytic approach uses the information stored in object and sensor models to analytically predict the appearance of an object from various viewpoints. Alternatively, it is possible to generate images of objects under specific sensor conditions and analyze the synthetic images. Both techniques have advantages and disadvantages that are discussed more completely in the next two sections.

The appearance of an object varies with respect to the sensor used. The appearance of an object with respect to a sensor is characterized by means of the features that can be extracted from the sensor image. Hence, each sensor model includes a feature set, and a collection of image processing operators that are used to extract the features.

The appearance of an object also varies with respect to the relative geometry between sensor and object, which can be referred to as the viewpoint. Potentially, there are six degrees of freedom in viewpoint, each of which spans an infinite number of parameter values. Clearly, exhaustive computation of all possible appearances is impossible. To make the set of possible appearances manageable, similar appearances are grouped into sets called aspects. Formally, an aspect is a class of topologically equivalent views of an object [17]. However, since different sensors detect different features, the formal definition of an aspect is usually relaxed to be a class of appearances that are equivalent with respect to a feature set.

A substantial amount of work has been performed on deriving methods for analytically determining the collection of aspects of an object. For example, Plantinga and Dyer [19] compute the exact set of aspects for polyhedra under either orthographic projection or perspective projection, using the definition of aspects as topologically equivalent views. Kriegman and Ponce [18] compute the exact set of aspects for solids of revolution under orthographic projection, again using the definition of aspects as topologically equivalent views. Chen and Freeman [6] determine the exact aspects for quadric-surfaced solids under perspective projection, where aspects are views with isomorphic line-junction graphs.

An alternative to the exact analytic computation of aspects is the *exhaustive approach*, in which viewpoints are sampled uniformly throughout the space of possible viewpoints, and then similar viewpoints are grouped together. An approach of this sort was used by Ikeuchi and Kanade [16], Hansen and Henderson [10], Flynn and Jain [7], and Sato, et al [20]. In each of these systems, the space of possible viewpoints is uniformly tessellated, and the object appearance is predicted from each viewpoint corresponding tr the center of a tessel. The fidelity of the sampling can be increased by subdividing each tessel. This approach is more general than the analytic approach, since the same procedure can be used independently of the sensor or feature set.

2.2.4 Generation of a Recognition Strategy

The result of the off-line, compilation phase of an appearance-based system is a strategy for object recognition. The strategy is often represented in the form of a tree that represents the sequence of operations to perform at each step of the recognition process. Since the generation of a strategy is performed off-line, it is possible to perform relatively expensive optimization.

There are many different computational approaches that can be employed for object recognition. Suetens, et al [22] is a recent survey of the range of approaches. A VAC can be constructed for any given approach. Consequently, there is no standard form for the recognition strategy output by a VAC; all that can be said is that the strategy consists of executable code.

The strategy generated on the basis of analysis of predicted appearances is repeatedly executed in an application. Since the strategy is executed repeatedly, optimizations performed in the off-line phase are essentially amortized over the many on-line executions.

3 A VAC Utilizing Analytic Feature Prediction

The VAC discussed in this section was designed to generate an object localization program for a bin-picking task, and was initially presented in [14], at which time the system was not fully automatic. Further research has led to complete automation of program generation [15], as well as optimization of the resulting code [12]. This section will present the complete system.The inputs to the VAC consist of an object model, specifying geometric and photometric characteristics of the object, and a sensor model, specifying the sensor characteristics necessary for predicting object appearances and feature variations. The output consists of a recognition strategy in the form of an interpretation tree.

A localization task is solved in two phases, as is characteristic of appearance-based systems. In the first, off-line phase, the object and sensor models are used to predict object appearances and the variation in detectable features. The result of the first phase is a recognition program for the given task. The second, on-line phase consists of applying the generated program to the actual task.

A 3d object can yield an infinite number of 2D appearances for a given sensor, resulting from changes in viewpoint. But, for any given sensor, there are a finite number of qualitatively different characteristic appearances that can be termed *aspects*. An input image can be classified into an instance of an aspect based on the ranges of the feature values that differentiate between aspects; this procedure is called aspect classification. Since aspects are essentially collections of viewpoints, aspect classification is equivalent to rough localization. More accurate localization can be performed by analyzing the shape change within an aspect, called linear shape change determination. The localization program generated by the system reflects the distinction between global and local shape changes by separating the processing into stages of aspect classification and linear shape change determination.

3.1 Explicit Object and Sensor Models

Computer vision systems can use many different types of sensors. A sensor can be considered to be a transducer that transforms object features into image features, and different sensors yield different image features. For example, a laser range finder detects the range and orientation of object surfaces, while edge-based binocular stereo yields the range computed by triangulation on detected edges. Table 1 summarizes various sensors in terms of detectable object features. A sensor model must specify the features detectable by the sensor.

The list of features describes the qualitative characteristics of a sensor. The quantitative characteristics are given by the *detectability* and *reliability* of each feature. Detectability specifies the conditions under which a given feature can be detected. Reliability specifies the expected error in the feature value. Both characteristics depend on the configuration of the object feature and the sensor.

The detectability of a feature by a given sensor depends on factors such as range, relative attitude, reflectivity and so on. In many applications, such as industrial workstations, many of the factors can be fixed, and relative attitude becomes the dominant factor. To consider relative attitude, fix the sensor coordinate system, and consider the relationship of a feature coordinate system with respect to it. The feature coordinate system is defined such that the z-axis is aligned with the surface normal; x and y axes are assumed to be defined arbitrarily. There are three degrees of freedom in orientation of feature coordinates with respect to sensor coordinates.

Consider a solid unit sphere, called the *orientation sphere*, or *o-sphere*, in which each relative orientation of the feature coordinate system corresponds to a point. The direction from the center of the sphere to the point defines the orientation of the feature z-axis. The rotation of feature coordinates around the z-axis corresponds to distance from the spherical surface to the center of the sphere; a point on the surface represents a feature coordinate system obtained by rotation around an axis perpendicular to the plane formed by the north pole, the center of the sphere, and the surface point itself. The north pole is taken to be the case in which feature coordinates and sensor coordinates are aligned. One o-sphere can be defined for each object feature, and is referred to as the feature configuration space. Figure 1 illustrates the concept.

Table 1: Summary of Sensors

Sensor	Vertex	Edge	Face	Active/ Passive
Edge detector	-	line	-	passive
Shape-from-shading	-	-	region	passive
Synthetic aperture radar	point	point/line	point	active
Time-of-flight range finder	-	-	region	active
Light-stripe range finder	-	-	region	active
Binocular stereo	-	line	-	passive
Trinocular stereo	-	line	-	passive
Photometric stereo	-	-	region	active
Polarimetric light detector	-	-	point	active

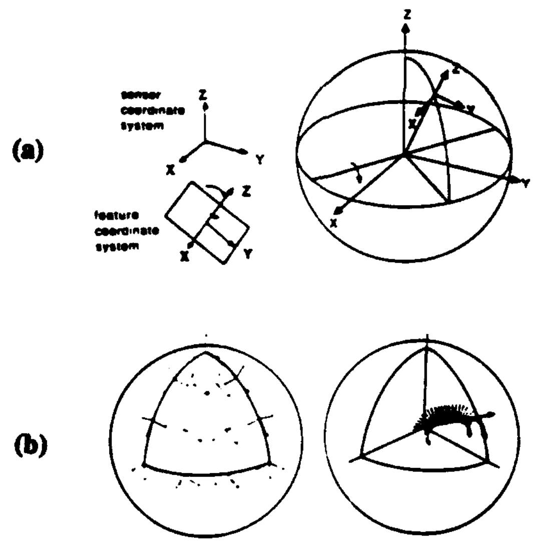


Figure 1: Feature configuration space.
(a) Relationship between sensor coordinates and feature coordinates. (b) Feature coordinates as points on the o-sphere. The bottom left drawing depicts the coordinates corresponding to points on the surface of the o-sphere, while the bottom right drawing depicts the coordinates along one axis of the o-sphere.

A sensor system consists of two components: an illuminant, and a detector. For a feature to be detected, it must be visible to both components. For a given feature, a separate configuration space can be defined for each sensor component. Within each configuration space, the configurations for which the feature is detectable can be easily defined by geometric constraints. For example, a plane is detectable by a conventional tv camera if the surface normal of the plane forms an obtuse angle with the camera line of sight. The detectability of a feature with respect to a given sensor is then the intersection of the detectable regions of the configuration spaces of each of the sensor components. An example of the detectability computation for a light-stripe range finder is shown in Figure 2.

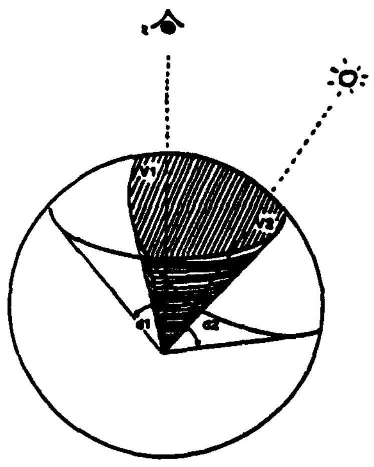


Figure 2: Detectability of a face for a light-stripe range finder. The detectable region is the intersection of the detectability of the illuminant and the detectability of the sensor.

For a given viewpoint, the appearance of an object with respect to a particular sensor can be defined by a list of the detectable object features, along with the values of parameters extracted from those features. A viewpoint corresponds to a single point in the configuration space of each feature of the object. A feature is detectable if and only if the point representing the sensor viewpoint lies in the detectable region of the feature configuration space, and if no other part of the object occludes the feature. These conditions can easily be checked using a geometric modeler. Figure 3 illustrates the process for a simple polyhedral object and a light-stripe range finder.

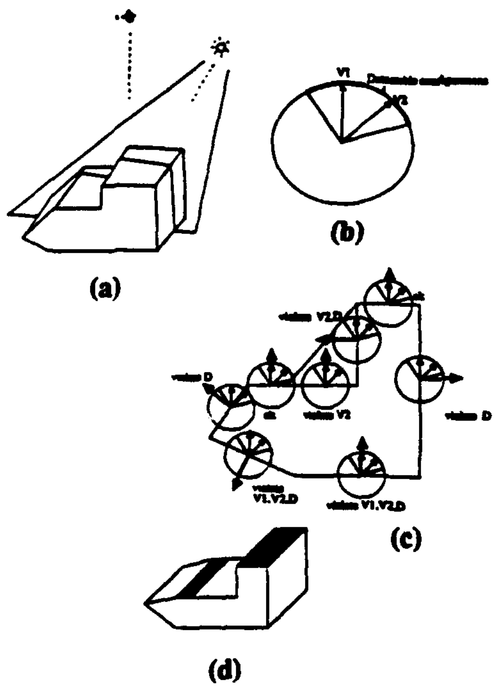


Figure 3: Use of detectability constraints.
(a) Polyhedral object and light-stripe range finder. (b) Detectability constraints of a face.
(c) Application of detectability constraints. (d) Detectable faces.

3.2 Predict and Analyze Appearances

The techniques presented above make it possible to analytically determine the detectability of any feature from any viewpoint. The combination of features, along with predicted feature values, defines the appearance of the object. Next, the capability to predict appearances must be used to determine the aspects of the object.

For many feature sets, analytic approaches to determining the aspects of an object have been derived ([6], [18],[19]). However, each approach is specialized for a specific feature set and a limited collection of surface types. An alternative approach, known as the *exhaustive approach*, is to examine a representative set of viewpoints around the object; this approach is independent of the feature set and surface type. As the sample set grows and the spacing between samples decreases, the results from the exhaustive approach will agree arbitrarily closely with the analytic results.

The exhaustive approach is relatively easy to implement, especially for cases in which the distance between sensor and object is assumed fixed. Then, all possible viewpoints can be represented as points on the surface of a sphere centered on the object. A tessellation of the sphere using a geodesic dome which divides the sphere into many small spherical triangles yields a nearly uniform sampling of viewpoints. The triangles can be subdivided repeatedly to yield any desired level of sampling resolution. At the center of each spherical triangle, the detectability of each feature can be computed as outlined above.

Aspects can be selected in many different ways, depending upon the features being considered. For example, aspects can be defined as collections of viewpoints for which the same set of features are visible. Alternatively, as used here, aspects can be defined on the basis of detectable faces. Consider an object with N faces (planar or curved) S_1, $S_2,...,S_N$ and define the face-label $X = (X_1, X_2,...,X_N)$, where $X_i = 1$ or 0 according to whether or not face S_i is detectable. Viewpoints with identical face labels are grouped together into aspects. For each aspect, a representative attitude is selected and used to calculate representative feature values. Each aspect can be characterized by the feature values of the representative attitude. Further, the ranges of feature values can be obtained by examining the range of values of the features of each of the viewpoints constituting an aspect. Figure 4 illustrates the process of view generation and aspect selection.

3.3 Generation of Recognition Strategy

The generation of a recognition strategy depends to some extent upon the sensor used, or at least upon the features used. In this section, results are presented for sensors which produce dense range maps.

3.3.1 Aspect Classification

Aspect classification is the process of classifying an input image into an instance of an aspect. Since an aspect represents a contiguous set of viewpoints, aspect classification is equivalent to rough localization. The parameters of object pose determined through aspect classification also provide good starting parameters for the stage of linear shape change determination that follows.

One way of performing aspect classification is to extract feature values from the input image and compare this set of values to the stored value ranges that characterize aspects. This approach may be very inefficient, however, since only a few of the features may be needed to perform classification, yet all are computed. A more cost-effective approach is to determine the computational cost of each feature, and then determine a discriminating set of features that minimizes the expected cost of classification.

A *classification tree*, or *decision tree*, is a tree in which each node represents a collection of classes and an associated test, and arcs represent the possible results of a test. Leaf nodes represents the final results of classification. Using a classification tree, a classification is performed by traversing the tree from the root to a leaf. A classification tree provides a convenient framework for optimizing the aspect classification process, since using a classification tree permits tests (and corresponding computations of feature values) to be performed sequentially.

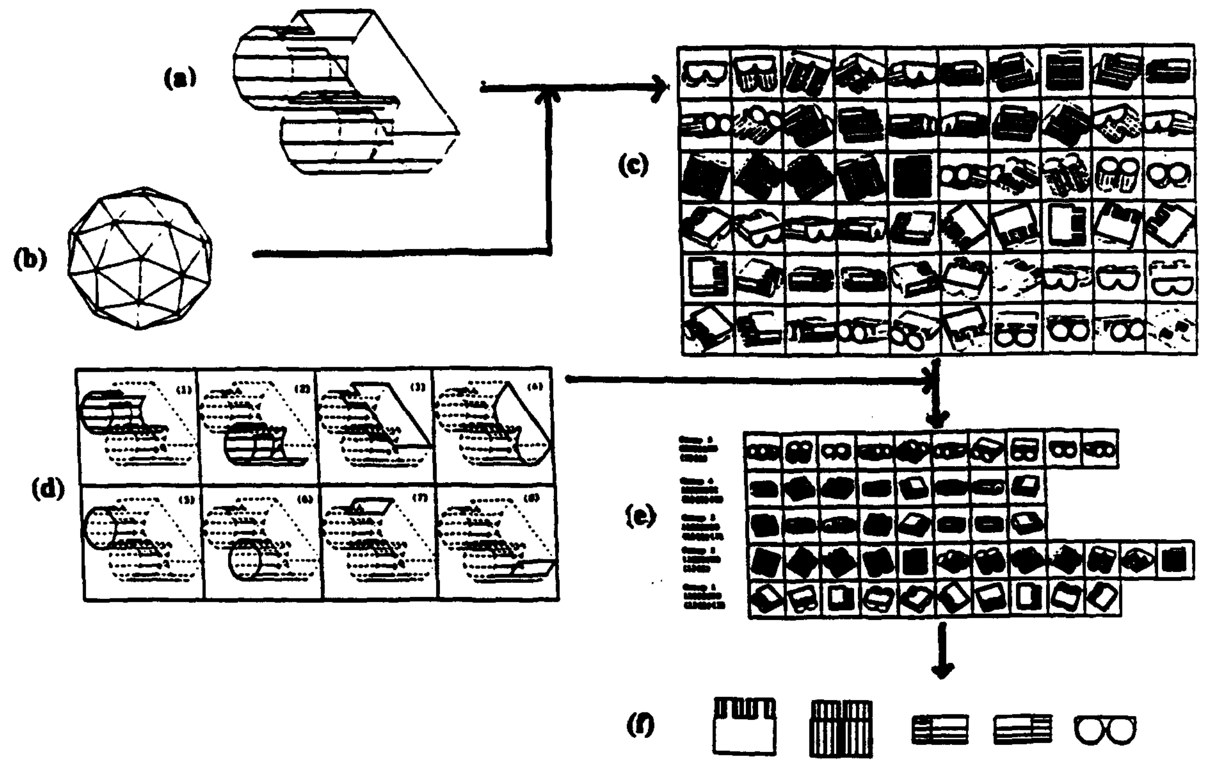


Figure 4: Extraction of aspects.
(a) Geometric model of an object. (b) The Gaussian sphere is tessellated into sixty triangles to represent viewpoints sampled. (c) Sixty computed appearances. Faces surrounded with bold lines are detectable by photometric stereo. (d) Eight component faces to be used for shape labeling. (e) The five aspects obtained through classification by shape label. (f) Representative attitudes, one for each aspect.

Aspect classification trees can be used to optimize the classification process in the following way. In the off-line stage of processing, the entire set of possible aspect classification trees is examined systematically, and the minimum cost classification tree is identified and saved; this tree stores feature identifiers and test values at each node.

A path from the root of a classification tree to a leaf represents a complete classification operation. Computing the cost of a single path is straightforward. Each test requires a feature to be computed, and each such computation incurs a computational cost. Each node in the classification tree is assigned the cost of computing the feature needed for the corresponding test. The cost of a path is the sum of the costs of the intermediate nodes.

A classification tree contains many paths from the root to the leaf nodes, and different paths may be taken with different frequencies. Therefore, the cost of a classification tree is defined to be the expected cost of a classification; that is, the average cost taken over all possible inputs. The expected cost can be computed by weighting the cost at each node by the proportion of the sample population that will pass through the node. The cost of every node in the tree is summed and divided by the population size to yield the overall cost. Figure 5 illustrates the method for computing the cost of an individual classification and the overall cost of a classification tree.

Finding a minimum cost classification tree can be formalized as a search problem over another kind of tree, called a *strategy tree*. A strategy tree for aspect classification is a tree in which each path from the root to a leaf represents a complete strategy for classification; that is, each path in a strategy tree can be expanded into a classification tree. Therefore, finding the least cost classification tree is equivalent to finding the least cost path from root to a leaf in the strategy tree.

In a strategy tree, each node contains the results of applying a test using a given feature, while arcs represent the tests. Each arc is labeled with the product of the cost of the feature and the expected number of samples to which the test

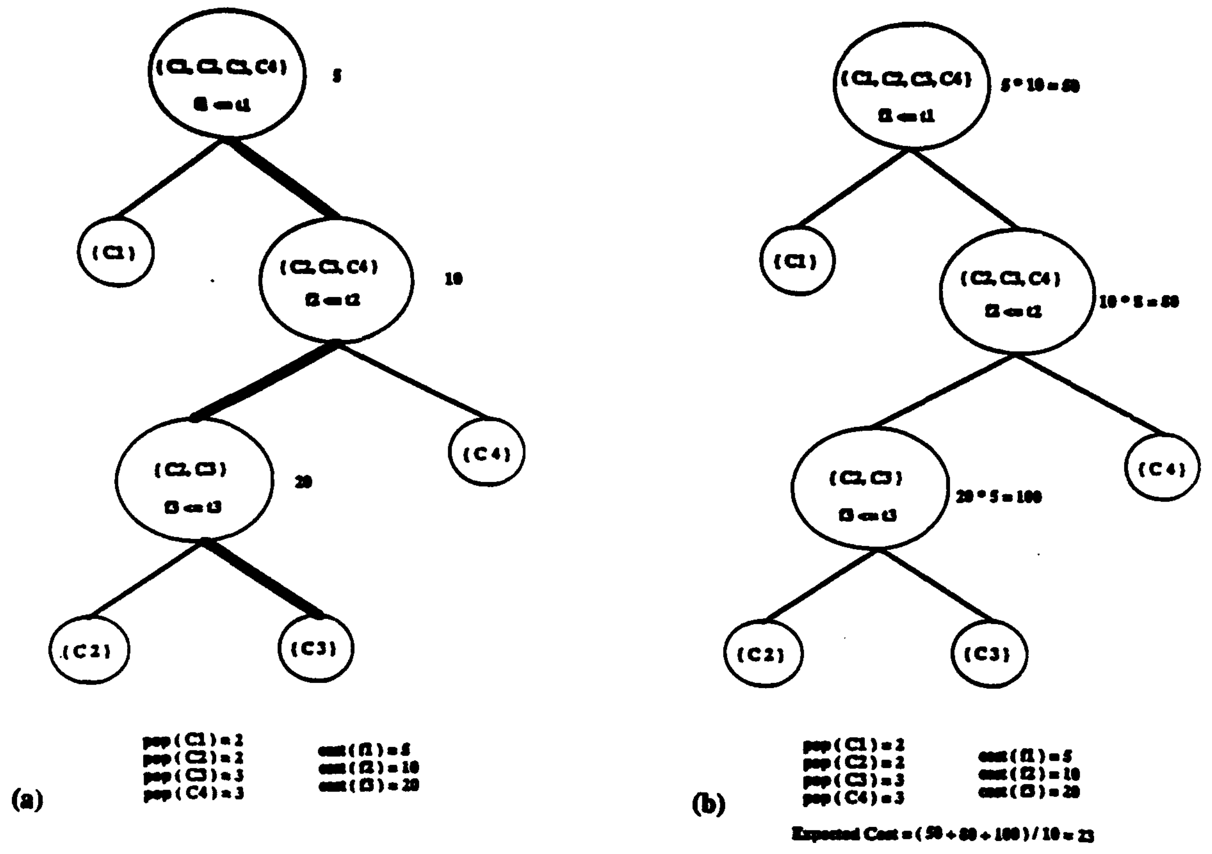


Figure 5: Cost computations.
(a) Cost of classification (b) Expected cost of a classification tree

will be applied. An arc is present when a feature can be used to break up a set into smaller sets. At the leaf nodes, all the constituent sets should be singletons, unless the feature set is incapable of distinguishing some of the input classes.

illustrates a strategy tree for a simple case consisting of 4 classes, and 3 features. At the root, all the classes are grouped into a single set. An arc is present for every computable feature which can reduce the set size, so there are three arcs at the root. The darkened path in the tree is the minimum cost path, and expands into the classification tree shown in Figure 5.

There are cases for which is not possible, given the set of features, to distinguish two classes. Indistinguishable classes are referred to as *congruent classes*, and the corresponding nodes as *congruent nodes*. Since the classes in an aspect classification tree represent aspects of some object, the existence of congruent classes means that the corresponding aspects cannot be distinguished with the available features; such aspects are referred to as *congruent aspects*. Congruent aspects do not represent a failure of the search procedure, but rather indicate a fundamental limitation of the feature set. In many cases, the linear shape change determination step corrects for ambiguous aspect classification and determines the correct object pose.

3.3.2 Linear Shape Change Determination

The aspect classification step results in the classification of an input image as an instance of an aspect. Since an aspect consists of a contiguous collection of views of an object, aspect classification is equivalent to rough localization; the possible collection of object poses are limited to those consistent with the observed aspect.

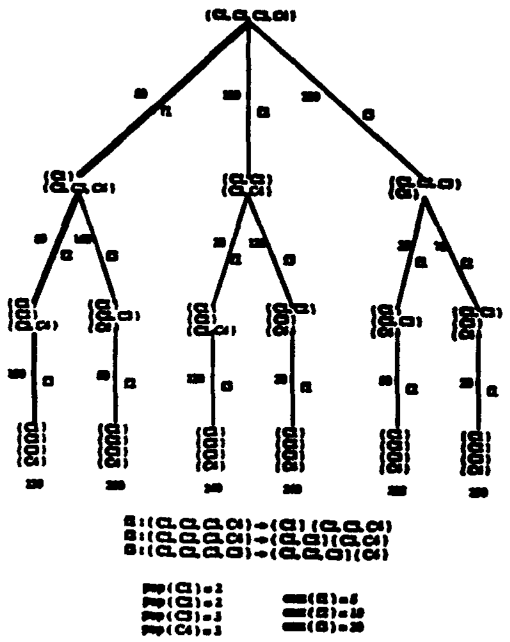


Figure 6: Strategy tree

The next step in the localization process determines the exact pose, given the initial estimate obtained from aspect classification. The same set of features is visible throughout an aspect, so no non-linear events such as the appearance or disappearance of a feature occur. Therefore, the second step consists of determining the exact pose within an aspect, subject only to linear changes (rotation and translation) of features; this step in the localization process is known as *linear shape change determination* (LSCD).

One way to perform LSCD is to utilize a model-based approach in which image features are matched to model features, a pose is hypothesized and used to predict locations of features in the image, and the pose is refined by computing the error in predictions and observations and updating the pose appropriately. In most model-based systems, the matching stage is very difficult, since all possible matches between image and model features must be investigated. Since aspect classification has been performed, however, the correspondence between some set of image and model features has already been established. In particular, the assumptions underlying the LSCD method presented here are:

- the correspondences between model and image faces are known;
- the correspondences between model and image edges are unknown.

The aspect classification strategy presented in the previous subsection was encoded in the form of a tree, the classification tree. Each leaf node of the tree represents an aspect or collection of congruent aspects, and for each leaf node a different LSCD strategy may be appropriate. The VAC presented here computes a separate LSCD strategy for each leaf node of the aspect classification tree. The steps in each LSCD strategy are encoded as nodes that are attached to the leaf nodes of the aspect classification tree. Although the particular computational procedures vary between aspects, the same steps are followed in the same order for each aspect:

1. determine the coordinate system of the primal face (the visible face with the largest 3D area):

   1.1. determine the origin of the primal face;

   1.2. determine the z-axis orientation of the primal face;

1.3. determine the x-axis orientation of the primal face;

2. estimate the body coordinate system;

3. establish correspondences between image and model edges;

4. recover exact body coordinates by numerical minimization.

The LSCD strategy is determined off-line through the following steps:

1. For each aspect, the visible face with the largest 3D area is selected as the *primal face*;

2. Each primal face is analyzed, and a method for defining the face coordinate system is determined. Separate nodes are attached to the classification tree which define the exact procedures used in each individual step of determining the origin, z-axis, and x-axis of the primal face.

3. Given the estimated coordinate system of the primal face and the transformation between primal face coordinates and model coordinates, the body coordinate system can be estimated with respect to the sensor coordinates. This is encoded as a separate node of the tree.

4. Knowing the object aspect and a rough estimate of the body coordinate system enables the prediction of the location of model edges in the image. The predicted edges can then be matched to observed edges. This process is encoded as a separate node of the tree.

5. A fine-tuning procedure is used to determine the exact body coordinates by adjusting the estimated body coordinates so that image edges exactly match predicted model edges. An exact match is not possible, so the procedure finds the body coordinates that minimizes the error between predicted and observed edge locations. This procedure is encoded as a separate node of the tree.

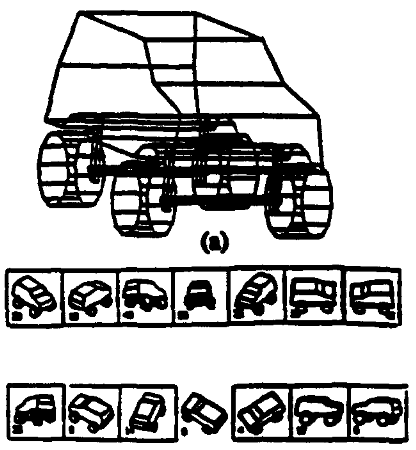
3.3.3 The Interpretation Tree

The overall strategy for object localization is encoded in the form of a tree, the *interpretation tree*. The top part of the interpretation tree consists of the aspect classification tree, and consists of directions for a series of feature value computations and tests that result in the classification of an input image into an instance of an aspect. The bottom part of the interpretation tree is six nodes deep, and consists of the steps in the LSCD strategy that are appropriate for each aspect. Figure 7 and Figure 8 illustrate the interpretation trees for two objects: a toy car, and an L-shaped polyhedron, respectively.

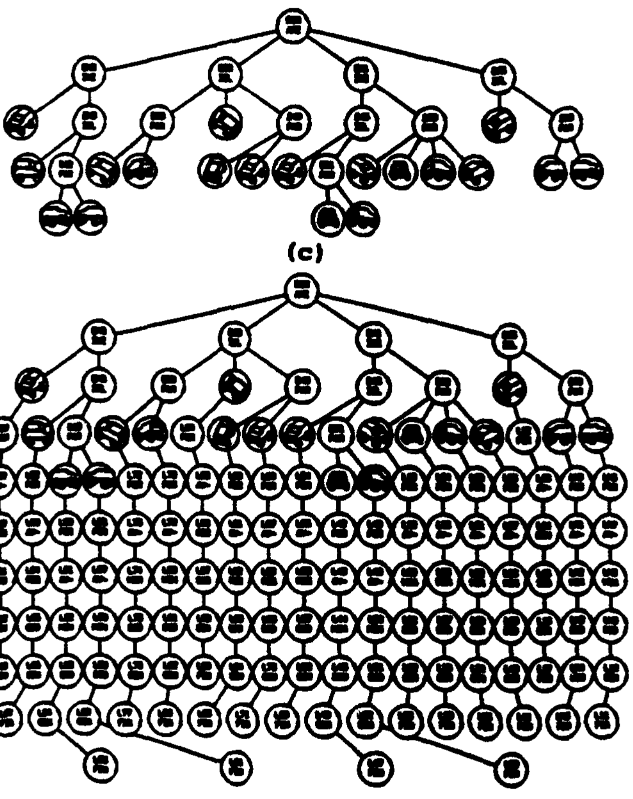
3.4 Run-time Execution

Each of the procedures represented by a node of an interpretation tree corresponds to an executable object stored in a program library. During the off-line phase, the creation of a node of the interpretation tree is accompanied by the instantiation of an executable object from the program library, and the insertion of the object at the node. During the on-line phase, these instantiated objects are executed in order to perform object recognition. Message-passing is used for communication between objects.

For each execution of the object recognition strategy during the on-line phase, a preprocessed image is passed to the root of the interpretation tree and the object stored at the root is invoked. The execution of the root object results in the computation of some feature value and the application of a test. The outcome of the test results in the preprocessed image being passed to the appropriate node at the next level of the tree, and the corresponding stored object is invoked. The sequence of message-passing and object invocation proceeds until a leaf node is reached, indicating that processing is complete. The results of processing are then passed back up the tree and returned from the root.


(a)

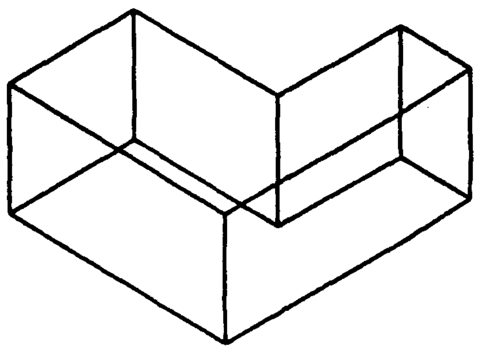
(b)

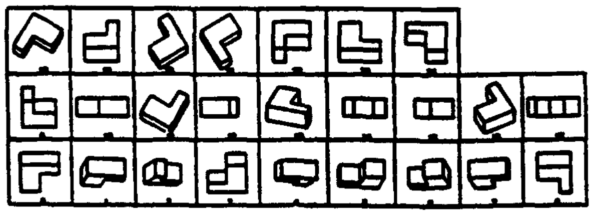

(c)

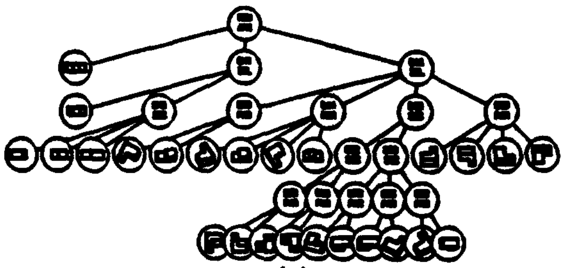
(d)

Figure 7: Generation of an object recognition strategy for a toy car.
(a) Object model. (b) Object aspects
(c) Aspect classification tree (d) Complete interpretation tree

At congruent nodes of the classification part of the tree, no unique aspect is identified. Instead, several aspects are determined to be possible. For each congruent aspect, the LSCD part of the tree is executed, and the results returned are compared by the congruent node; the aspect yielding the minimum error is selected as the correct interpretation and this result is passed back up to the root of the tree.


(a)


(b)


(c)

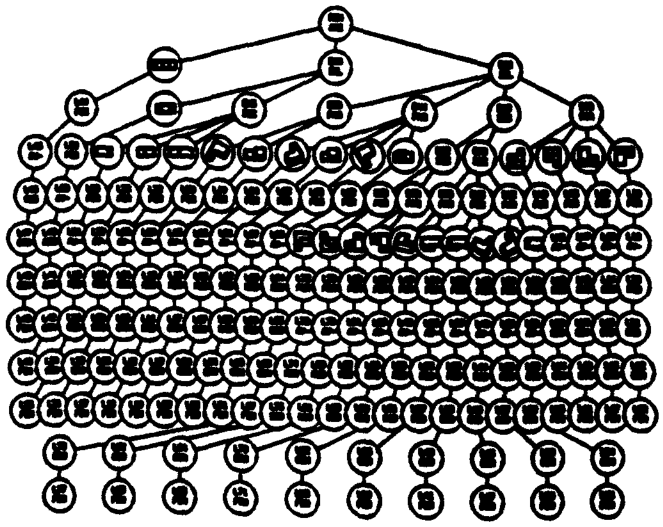

(d)

Figure 8: Generation of an object recognition strategy for an L-shaped polyhedron. (a) Object model. (b) Object aspects. (c) Aspect classification tree. (d) Complete interpretation tree.

3.5 Experiments

In this section, we illustrate the execution of the compiled strategy. As stated previously, the VAC presented here was constructed to utilize sensors that produce dense range images; that is, the features used are those that can be determined given an input range image. In order to show the sensor independence of the VAC, the results are demonstrated using two different range sensors: dual photometric stereo [13], and an Erim laser range finder [11].

3.5.1 Dual Photometric Stereo: Toy Car

The complete interpretation tree for the toy car is presented in Figure 7. The toy car was placed in a scene on top of a pile of other objects. The input car scene is shown in Figure 9. Preprocessing stages results in the computation of:

- A needle map containing the gradient space values at each pixel.
- An edge map.
- A label map indicating the region to which each pixel belongs.

Figure 9: Input scene for dual photometric stereo experiment

The aspect classification steps performed are illustrated by the black nodes shown in the interpretation tree of Figure 10. Starting below the node at which the aspect is identified, the LSCD processing begins. The first node determines the mass center of the target region and declares that position to be the origin of the face coordinate system. The next node determines the average surface orientation of the target region and declares that to be the orientation of the z-axis of the primal face coordinate system.

The next two nodes complete the determination of the face coordinate system and estimate the body coordinate system. Figure 11 illustrates the estimated body coordinates overlaid on the input image.

The next step in the process is the determination of the edge pairs and fine-tuning of the object pose. Figure 12 illustrates the results of these processes.

3.5.2 Erim Laser Range Finder: L-Shaped Polyhedron

The complete interpretation tree for the L-shaped polyhedron is presented in Figure 8. The L-shaped polyhedron was placed in a scene and an Erim image obtained and presented to the root of the interpretation tree. The results are pre-

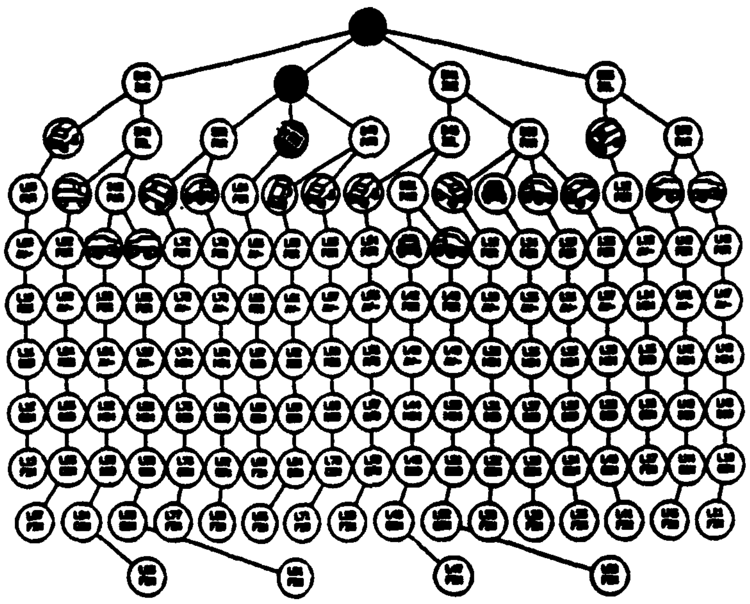


Figure 10: Execution of aspect classification tree. Black nodes indicate path followed.

Figure 11: Estimated body coordinates overlaid on input scene.

sented in Figure 13. The results are interesting in that a congruent node was encountered. The node contained two aspects, and the figure shows the two possible interpretations overlaid on the input image. Based on a comparison of the resulting errors, the correct aspect and pose were determined.

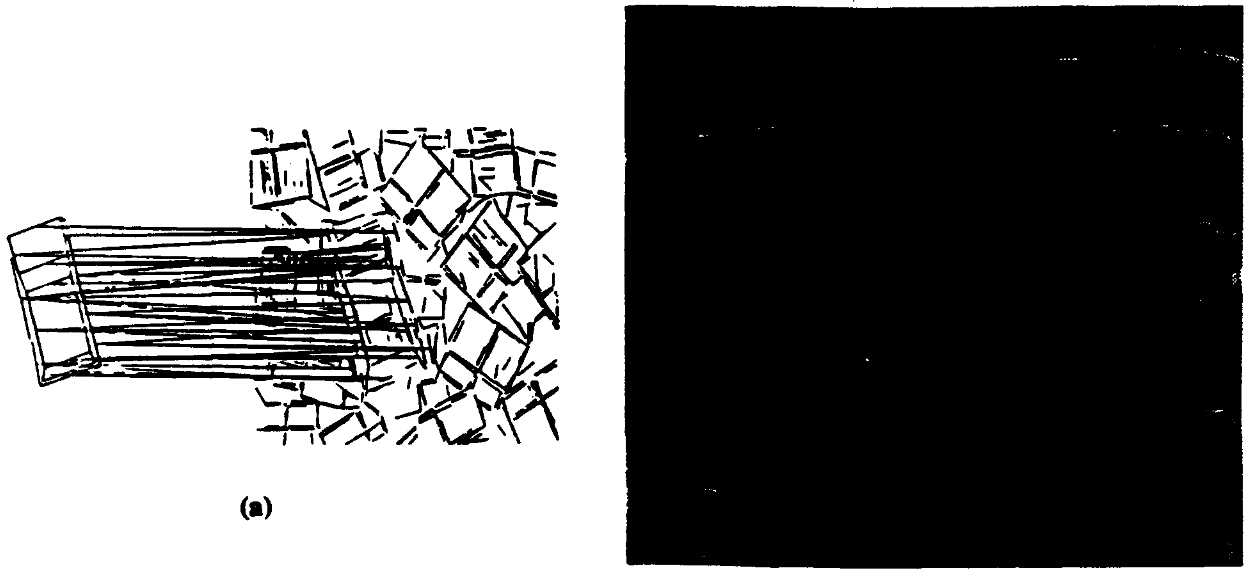

(a) (b)

Figure 12: Final determination of object pose.
(a) Correspondences between model and image edges. (b) Final pose overlaid on input scene.

4 A VAC Utilizing Image Synthesis for Prediction

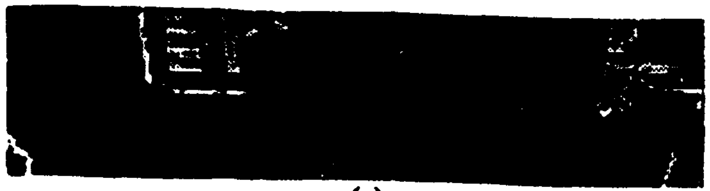
The appearance-based system of Sato, et al [20] was built for the purpose of recognizing specular objects. Specular objects pose a special problem for computer vision. Specularities are often the most prominent image features, and yet contain no brightness variations which can be used for edge detection or 3D shape analysis. Moreover, specular features may appear, disappear, or change shape abruptly with small variations in viewpoint. The presence of a specularity requires a precise configuration of illuminant, surface normal, and sensor, and therefore provide a powerful constraint on the underlying surface geometry. However, the constraint is purely local, and does little to constrain the object pose.

Specular reflections are found in nearly every imaging scenario. Metal, glass, plastic, and many other materials are highly specular. In addition to optical images, there are other imaging systems that are based on specular reflection. For example, radar is based on specular reflection, as are ultrasonic underwater imaging and medical sonography. Therefore, it is important to establish techniques to recognize objects using specular images.

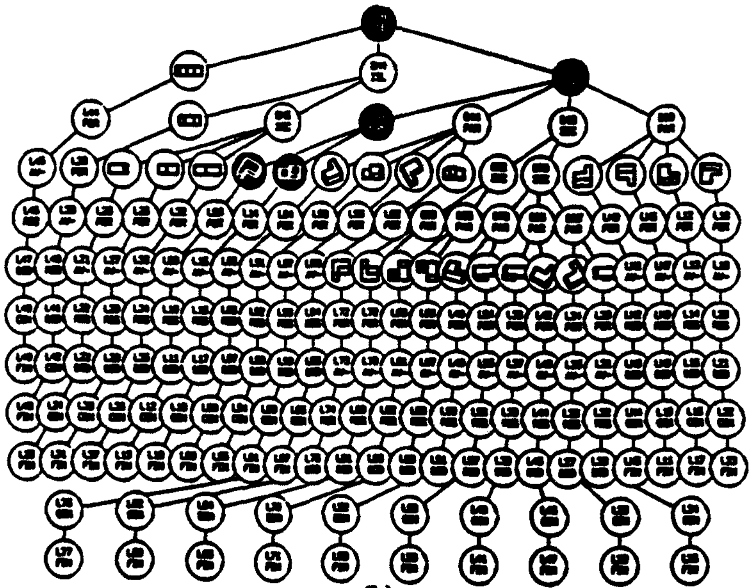
4.1 Explicit Object and Sensor Models

An analytic approach to appearance-based vision is impractical in the domain of specular objects. Because of interreflections between shiny surfaces, analytic prediction of specularities is extremely difficult. An alternative is the use of a sensor simulator, which generates object appearances based on both object and sensor models.

Sensor simulators are very similar to ray tracers of computer graphics. A 3D scene is described by a geometric modeler. A sensor simulator traces the path of light rays from pixels into the scene. Every time a ray hits an object surface, the ratios of reflected and transmitted energy are computed on the basis of the surface reflectance function and coefficient of refraction, respectively. The sensor simulator then traces both the reflected and refracted rays. When a ray reaches a light source, the energy emitted by the source is specified by the sensor model. The incident energies of all the rays towards a pixel are summed to determine the brightness value at the pixel and therefore predict the object appearance at the pixel.

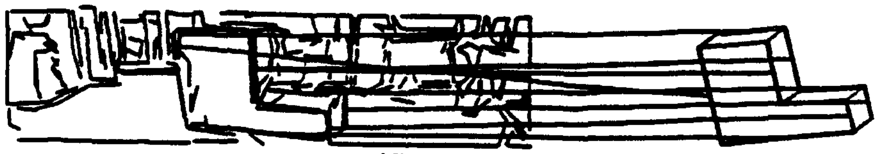


(a)

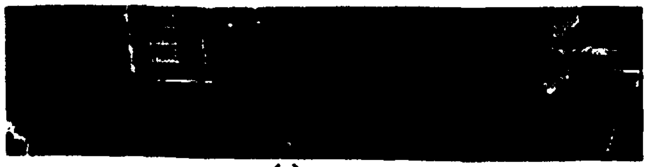


(b)

(c)



(d)



(e)

Figure 13: Erim Laser Range Finder Experiment.
(a) Input scene. (b) Results of aspect classification. (c) Two possible poses.
(d) Edge correspondences. (e) Resultant pose.

One critical difference between a sensor simulator and a ray tracer is that sensor simulators maintain the symbolic correspondences between regions of the image and the object surfaces underlying the regions. In the case of specular objects, this correspondence permits the analysis of which object surfaces produce strong, stable specularities, and from which directions specularities are visible on each surface.

Specular features are characterized by strong, distinct, and saturated brightness. In many cases, specularities can be extracted by the simple procedure of binary thresholding. Some specularities are easier to detect than others, however. In general, the size of a feature determines the ease with which it can be detected. For example, elongated specularities on a cylindrical surface are easy to detect, while a specular spot on a small sphere is difficult to detect. Thus, the detectability of a specular feature is related to the 3D shape of the surface underlying the feature.

While a specular feature might be easily detected, it could be quite unstable; that is, the specularity might be visible only over a small range of viewpoints, and a slight change in viewpoint could cause it to disappear. Such specularities are termed *unstable*, and are poor choices for use in recognition. The stability of a specular feature is related to the size of the collection of viewpoints from which the feature can be detected.

To make the concept of stability more clear, consider a co-located camera and light source. Specular reflections are detectable when the object surfaces are nearly perpendicular to the camera line-of-sight. Consider the motion of the camera/light system around the surface of a sphere centered on an object, with the line of sight always toward the center of the sphere. If a small specular sphere is being imaged, a specular spot will be observed, and will continue to be observed for all viewpoints; hence the specularity arising from a spherical surface is extremely stable. Now consider a cube being imaged. Each planar surface only yields a specularity when the line of sight is perpendicular to the surface, and the specularity disappears for small changes in viewpoint; hence, specularities from planar surfaces are unstable. The area on the viewing sphere corresponding to detectable viewpoints is a measure of stability of a specular feature.

Figure 14 illustrates the detectability and stability for specular features over four different surface types: planar, cylindrical, conical, and elliptical. As can be seen in the figure, planar surfaces have easily detectable specularities that are low in stability, while spherical surfaces have low detectability but high stability. Cylindrical and conical surfaces fall somewhere in between.

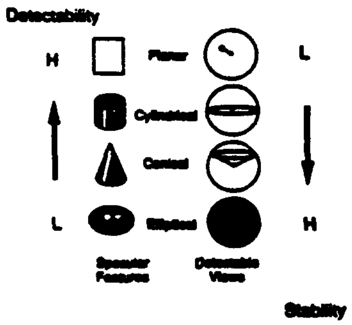


Figure 14: Detectability and stability of specular features

4.2 Predict and Analyze Appearances

The system employs an exhaustive method for appearance analysis. Using a sensor simulator, the system generates synthetic images for a representative collection of viewpoints obtained by tessellating the unit sphere. The viewpoints are then grouped into aspects on the basis of similar feature sets.

Assuming that range to the object is constant, all possible viewing directions can be represented as points on the unit sphere. A geodesic partition of the viewing sphere uniformly tessellates the sphere into small triangles. The center of each triangle is chosen to represent a viewing direction. Triangle can be further subdivided into smaller triangles to make the sampling as fine as desired. The sensor simulator is then used to generate synthetic images at each representative viewpoint. Figure 15 shows some of the appearances generated for a simple object.

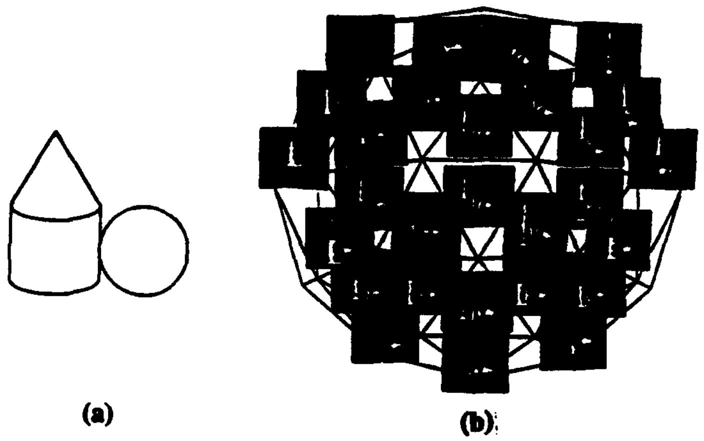


Figure 15: Specular images of a simple object.
(a) Object. (b) Synthesized sample images.

Each image is processed to extract specularities, and the data structure created by the sensor simulator is used to determine the primitive component underlying each specularity. For N primitive components $P_1, P_2,..., P_N$, a cell label can be defined as an N-tuple $(X_1, X_2,..., X_N)$ such that $X_i = 1$ or 0 according to whether or not component P_i gives rise to a detectable specularity at the viewpoint represented by the cell. Cells with identical cell labels are grouped together to form aspects. Thus, in the case of specular objects, aspects are defined with respect to detectable specularities. Geometrically, the process can be considered as follows. For each primitive component, a geodesic dome is constructed in which each cell is labeled according to whether the component gives rise to a specularity from that viewpoint. The geodesic domes for each component are then intersected, and each distinct region corresponds to an aspect. Figure 16 illustrates the selection of aspects.

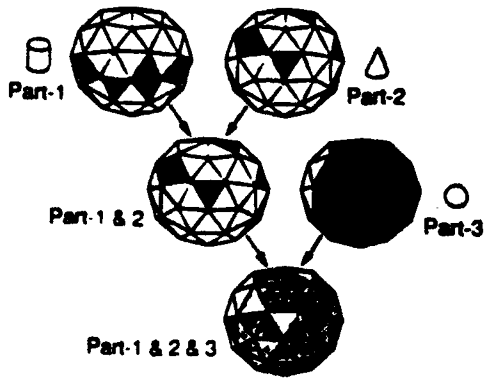


Figure 16: Aspect selection for the simple object.

4.3 Generate Recognition Strategy

As discussed above, specular features can vary in their characteristics of detectability and stability. For the purposes of object recognition, the system sorts specular features on the basis of detectability and stability in order to select the most effective features for aspect classification.

Detectability was defined above as the measure of the ease with which a specularity can be detected, and was related to the area of the specularity. The system uses as a measure of detectability the number of pixels of the largest appearance of a specularity, normalized by dividing by the area of the largest detected specularity.

Stability was defined as the area of the viewing sphere over which a given specularity is detected. In the case of a tessellated viewing sphere, this measure can be approximated by counting the number of cells within which the specularity is detected, normalized by the total number of cells.

An evaluation function is required to combine the measures of detectability and stability into a measure of overall feature utility. For each aspect, the features are ordered by decreasing utility. At run-time, matches are made in order of decreasing utility.

Aspect classification is equivalent to rough localization. Finer localization is difficult in the case of specular images because specular features change their shape drastically with small changes in viewpoint. Moreover, the exhaustive approach used in aspect determination may miss an unstable specularity that is only visible between two cells. Consequently, deformable template matching was selected as the procedure for fine localization.

Deformable template matching permits the template to deform according to certain constraints. An appearance is described as a combination of templates, each of which describes a specularity. The templates are interconnected conceptually by springs. The quality of match is measured by the sum of the internal deformation energy of the springs, and the external energy needed to fit each template to a real specularity. Thus, a deformable template can deform to find a match, even when a specularity changes shape or position. Moreover, matches can still be made even in the presence of accidental appearances or missing features.

A deformable template is prepared for each aspect using the appearance which is located at the center of the aspect. Specularities appear as spots or line segments, so each template consists of spots and line segments. Specular features are extracted from the central appearance. For an elongated feature, a line is fit to the feature and used to represent it in the template. For a spot feature, a point located at the center of the feature, is used to represent the feature in the template. A conceptual spring is located at each endpoint of a line feature, and at the point representing a spot feature. The spring energy is calculated from the displacement between the original and current location of the spring. Thus, the energy of a spot feature is a function of the displacement between the current and original position. For a line feature, the energy is a function of the displacement energy of the two endpoints.

4.4 Run-time Execution

Run-time execution is broken into two distinct stages: aspect classification and verification. In contrast to the system discussed in section 3, aspect classification does not uniquely classify an input image as an instance of an aspect. Rather, aspect classification is used to eliminate impossible aspects. Remaining aspects are input into the next stage, in which deformable template matching is used for verification.

4.4.1 Aspect Classification

Specular features can be very unstable. Small changes in viewpoint may cause a given specularity to appear, disappear, or change shape. Consequently, it is difficult to identify a single specularity, or the set of specularities that define an aspect, with complete confidence. Therefore, rather than employ a binary classification of an input image as an instance of an aspect, the run-time aspect classification system employs a continuous classification method based on the Dempster-Shafer methodology [21]. Figure 17 illustrates the classification method. For simplicity, the illustration is limited to aspects lying on a single great circle of the viewing sphere. Each match with a template from a single feature generates a likelihood distribution, in which a value close to 1 means that the corresponding aspect is very likely. Likelihood distributions from separate features are merged using Dempster-Shafer theory. As shown in the figure, each additional feature reduces the number of likely aspects and sharpens the peaks of the remaining ones. The likelihood values for impossible aspects decrease with each additional feature, while the likelihood values of possible aspects increase. After every the evidence from every available feature has been applied, the overall likelihood distribution may still contain several peaks, each of which represents a possible aspect classification for the input image.

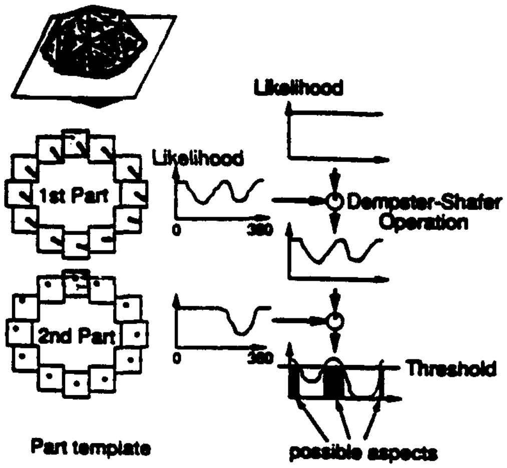


Figure 17: Aspect classification based on evidential reasoning

4.4.2 Verification

The verification process determines the correct aspect by matching the input image to the complete template for each of the possible aspects. Each template can move over the entire image to minimize the total energy. The total energy is comprised of a weighted sum of constraint energy, and potential energy:

$$E_{total} = W_{constraints} E_{constraints} + W_{potential} E_{potential}$$

Potential energy represents the energy of the position of the template, while constraint energy represents the energy of the relations between template components. Potential energy is readily visualized as the height of the template in a potential field defined by the detected specularities in the image. Constraint energy is modeled by springs connecting the template points to image feature points; as the template deforms, the springs stretch and the constraint energy increases. Figure 18 illustrates template matching.

An optimization procedure is used to find the energy minimum. To avoid getting trapped in local minima, some noise is added to the total energy. The global minimum energy for each template specifies the quality of fit of the input image to the template. The best match is chosen by comparing the minimum energies for each of the candidate aspects.

4.5 Experiments

In this section, the VAC for specular object recognition is applied to two different kinds of specular images: real optical specular images, and synthesized synthetic aperture radar (SAR) images.

4.5.1 Real Optical Images

For this experiment, a real toy airplane was constructed. The sensor used was a tv camera with a co-located light source. The sensor parameters were obtained by calibration. Figure 19 shows a real specular image of the toy airplane.

The object was modeled using the Vantage geometric modeler [2]. Object aspects were determined using the exhaus-

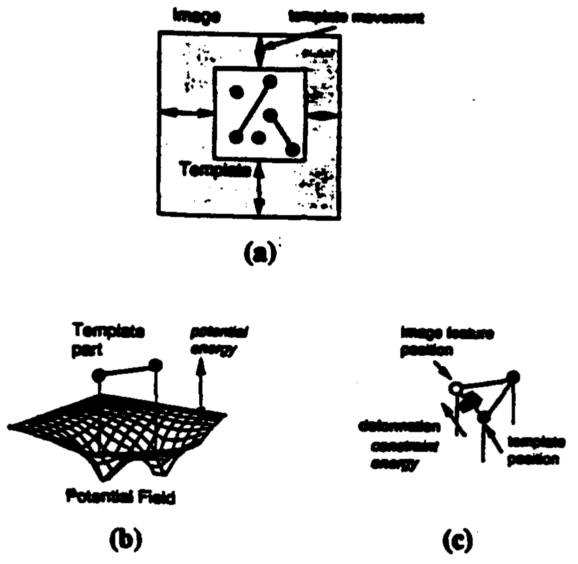
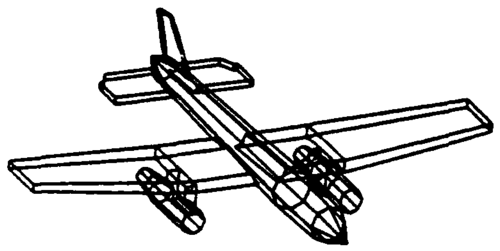


Figure 18: Deformable template matching.
(a) Template. (b) Potential energy. (c) Constraint energy.

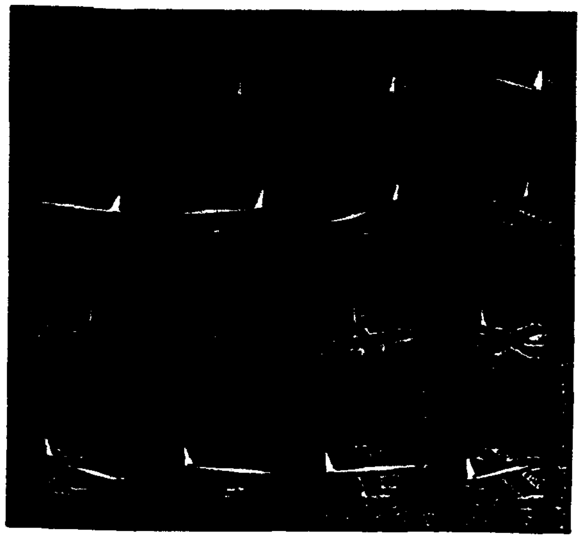
Figure 19: Real specular image of toy airplane.

tive method. Since man-made objects such as the airplane have only a few stable poses, an aspect map over the entire viewing sphere was not generated. Instead, only the appearances of the airplane from the equator of the viewing sphere were considered. Sample images were generated on the equator at 5° increments for a total of 72 samples. Appearances were generated using a sensor simulator [8]. Figure 20 illustrates the object model and some sample appearances.

The concentric arcs in Figure 21 correspond to the visibility maps for each primitive surface. The outermost arc corresponds to the detectable directions of the rear fuselage. The arc is unbroken - the part can be observed from all viewing directions. The top-left image in shows the set of possible appearances of specular features arising from the rear fuselage as a function of viewing direction. The other images correspond to other arcs of the visibility map. Some of the arcs in the map are broken; the missing arc regions correspond to viewpoints from which the primitive surface is not visible. The figure shows the features in their computed order of significance.

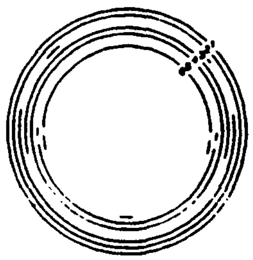


(a)

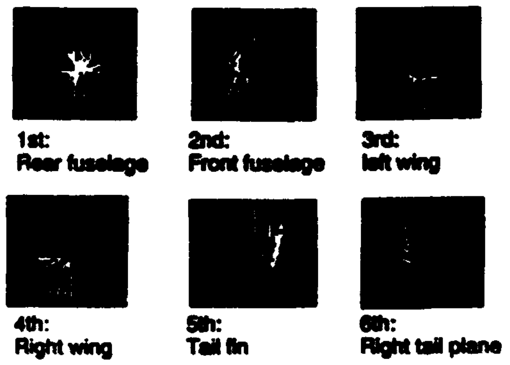


(b)

Figure 20: Model airplane and predicted specular appearances. (a) Airplane model. (b) Sample appearances.



(a)



(b)

Figure 21: Visibility and significance of features in optical experiment. (a) Visibility map. (b) Specular features in order of significance.

To test the resulting recognition program, a real specular image of the toy airplane was obtained and input to the system. The first step in the recognition process is aspect classification, in which possible aspects are searched by matching with partial templates. Figure 22 shows the input image and the results of matching to the first three partial templates. The figure clearly shows the narrowing of the likelihood distribution as additional matching is performed. The result of the aspect classification step was the selection of aspects at 45°, 115°, 125°, 165°, and 170°.

Following aspect classification, verification was performed using deformable template matching. Figure 23 illustrates the verification step. One template was used for each of the aspects selected. In each case the template changed its shape to match the specularities in the input image and converged to the shapes shown by the white lines superimposed on the copies of the real image in the figure. The minimum energy, and hence the best match, was obtained for the aspect at 170°.

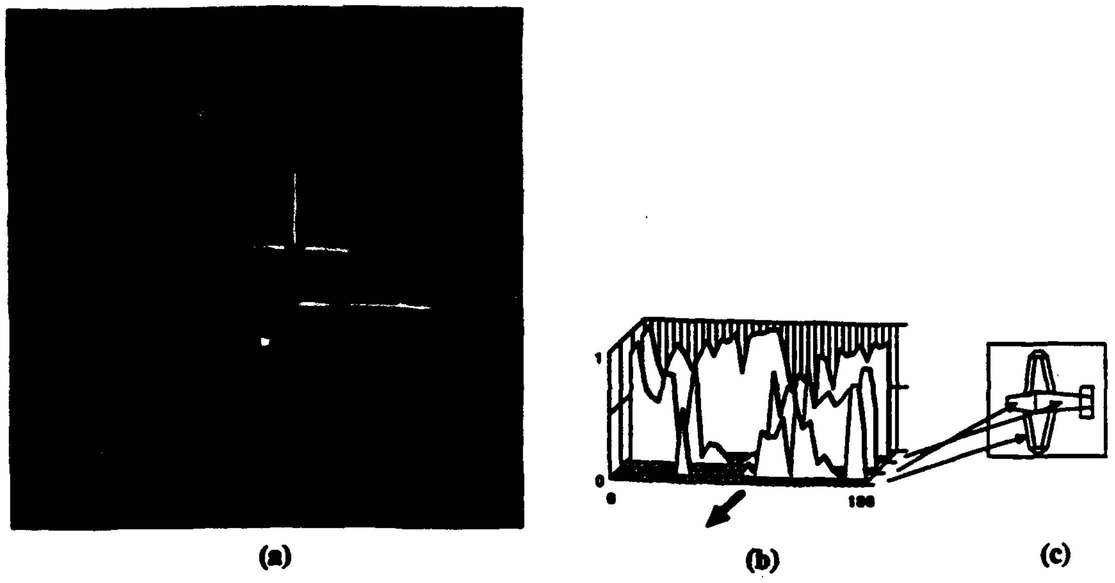


Figure 22: Aspect classification stage for optical experiment.
(a) Input optical image (b) Accumulation of evidence (c) Airplane parts.

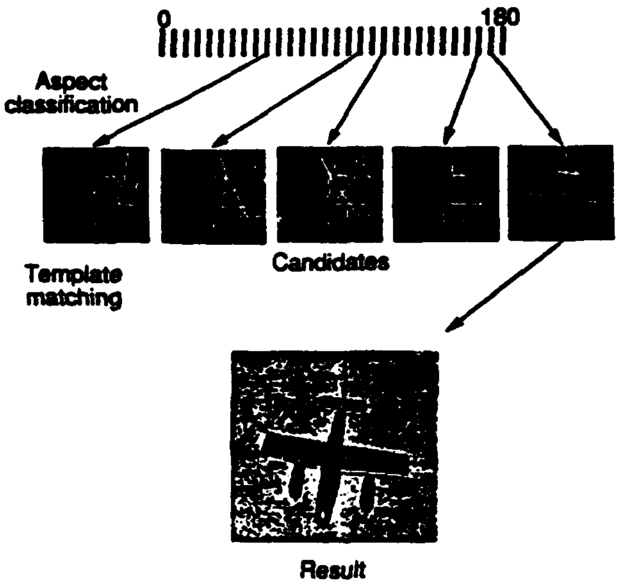


Figure 23: Verification stage for optical image.

4.5.2 Synthetic Aperture Radar Image

Synthetic aperture radar (SAR) is a flyable radar system that is often used on aircraft or satellites. SAR can produce very high-resolution two-dimensional images, and can detect details of targets, especially artificial structures. SAR images are based on specular reflections of radar waves, so image features in SAR are similar to specular features in optical image. In specific, the features are very sensitive to change in object orientation, change shape abruptly, and appear or disappear suddenly.

This experiment was performed using synthetic data only. A SAR simulator (SARSIM), developed by TASC [3], was used to generate synthetic SAR images. The simulator created not only brightness images, but also attribute images which specified the part of the object which caused each radar feature.

An airplane model was used for the object model that was similar to the airplane used in the optical experiment, but somewhat more complex. The output from a SAR sensor is a top-down view. By assuming that an airplane is parked on the ground, it was only necessary to consider the possible appearances of the airplane corresponding to various rotations about an axis perpendicular to the ground. Occlusions were not considered. Viewing directions were sampled every 10°. Figure 24 shows the airplane model and the sample appearances.

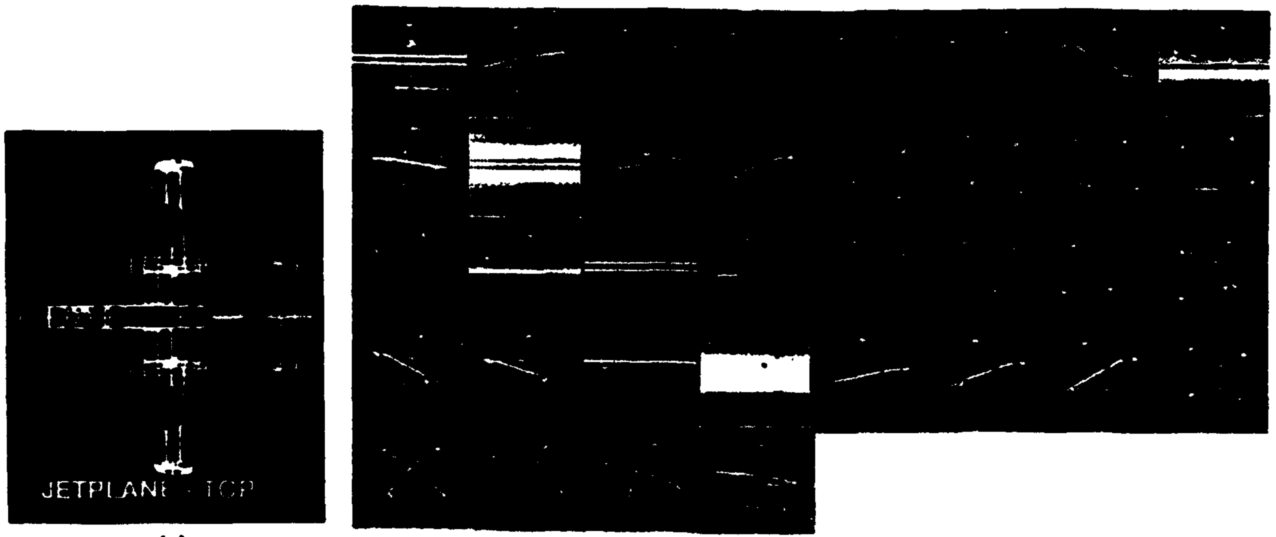


**Figure 24: Airplane model and predicted SAR images.
(a) Airplane model. (b) SAR images predicted by SARSIM.**

Aspects were determined using feature visibility. Figure 25 illustrates the visibility map for the airplane. Concentric arcs correspond to the detectable directions of each part; missing arc pieces indicate directions from which the part is not visible. The detectability and stability of each feature was evaluated from the simulated images, and the features were sorted in order of significance. The right side of the figure shows the appearances of each feature; the features are numbered in order of significance.

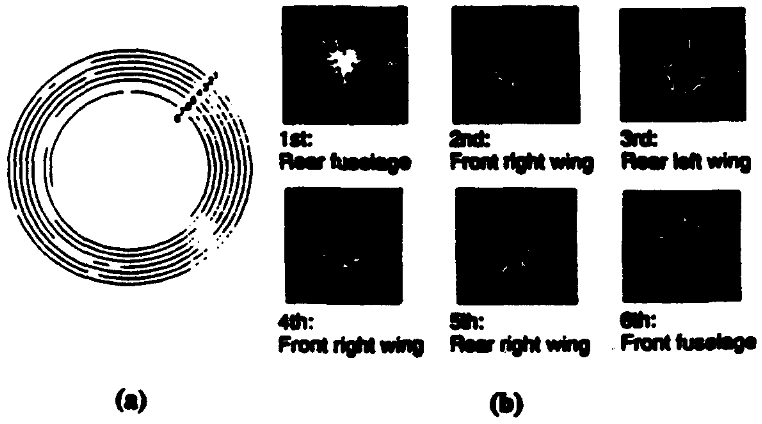


**Figure 25: Visibility and significance of features in SAR experiment.
(a) Visibility map. (b) SAR features in order of significance.**

Matching templates were generated based on the predicted features. Each template consisted of a collection of bright lines and bright spots. Since range values can be determined from SAR images, it was possible to normalize the size of aircraft. Hence, variation in object size was not considered.

A test image of the airplane was generated using SARSIM. The first stage of the run-time process was aspect classification, which reduced the number of possible aspects by matching with partial templates. Figure 26 illustrates the aspect classification process. The first match resulted in a broad likelihood distribution. After matching using all six effective features, a narrower distribution was obtained, which reduced the number of possible directions to five.

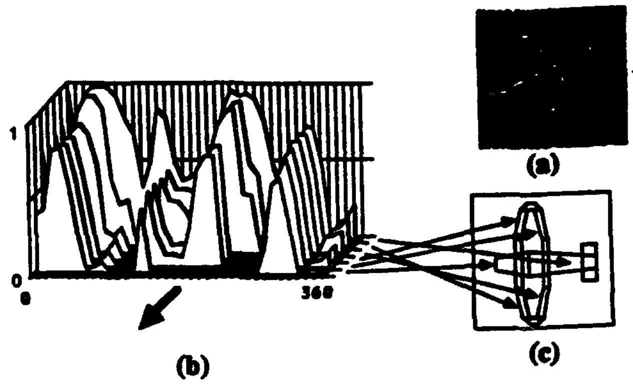


Figure 26: Aspect classification stage for SAR experiment.
(a) Input SAR image. (b) Accumulated likelihood distributions. (c) Aircraft parts.

The second stage of the run-time process was verification, in which deformable templates were matched to each of the candidate aspects. Figure 27 illustrates the verification process. The upper five images in the figure are copies of the input image. The superimposed white figures show the deformed templates at the minimum energy levels. The least energy of the five templates was obtained for the template at 30°. The result is shown in the lower image of the figure, with the outline of the aircraft superimposed over the original image.

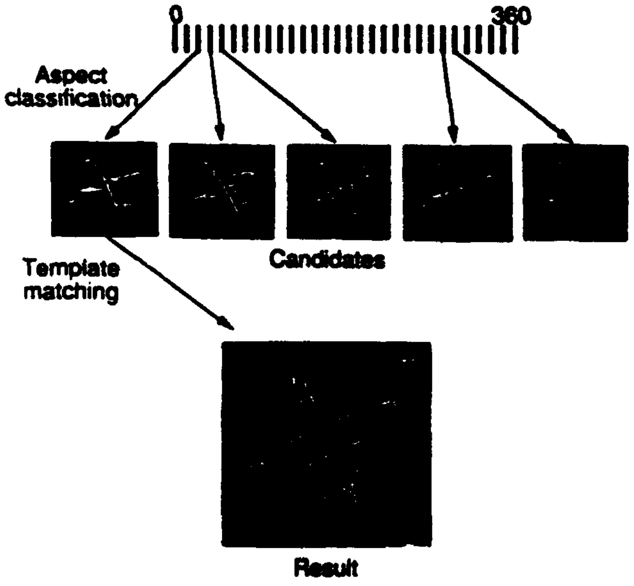


Figure 27: Verification stage for SAR experiment.

5 Summary

In this paper, we presented the paradigm of appearance-based vision, which is a paradigm for building object recognition systems. The paradigm is called appearance-based, since an integral step is the prediction and analysis of object appearances. An appearance-based system is called a vision algorithm compiler, or VAC. The input to a VAC is a set of object and sensor models, and the output is an executable object recognition program.

Appearance-based systems share four principle defining characteristics:

- two-stage process
  A VAC operates in two distinct stages. The first stage is performed off-line, and consists of the analysis of appearances and the generation of an object recognition program. The second stage is performed on-line, and consists of the execution of the previously generated program.
- explicit object and sensor models
  A VAC embodies an overall approach to object recognition that can be applied to a variety of objects and sensors. Therefore, explicit and exchangeable models are utilized. Sensor models specifies the features detectable by the sensor, along with procedures to compute the detectability and reliability of each feature. Object models include geometric and photometric properties of the objects.
- appearance prediction and analysis
  Objects are recognized based on their appearances in images. Therefore, the prediction and analysis of appearance is fundamental to generating competent object recognition programs. A VAC predicts appearances based on the information in object and sensor models. The range of appearances may be determined analytically or exhaustively, and the appearances may be predicted analytically or through image synthesis.
- strategy generation
  Each VAC may embody a different approach to object recognition. For example, the first VAC presented processed range data and performed final pose determination using minimization of edge location errors. The second VAC processed specular images and performed pose determination using deformable templates. However, the output of a VAC is an executable program for object recognition. Typically, the output program is optimized during the off-line stage; the optimization costs are paid back by cost-efficient execution of the on-line stage.

The history of computer vision research has consisted largely of research devoted to making vision systems work. As a result, powerful new methods have been developed and the vision systems of today are much more powerful and competent than those of the past. However, vision systems of today are no easier or cheaper to build than systems in the past. As a result, computer vision is not as widely applied as one might expect, due to the cost of systems.

Appearance-based vision provides one approach to making vision systems more cost-effective by providing a means of automatically generating object recognition systems. Rather than requiring time and effort from highly trained individuals, a VAC can generate a competent, cost-effective object recognition program given only object and sensor models. The example VACs presented in this paper demonstrated both the effectiveness and the flexibility of appearance-based vision.

Acknowledgements

The authors thank Robert W. Pinto and G. Stephen Zabele of The Analytic Science Corporation for offering the SAR simulator and giving valuable comments on the properties of SAR images. The authors also thank Takeo Kanade and the members of VASC (Vision and Autonomous System Center) of Carnegie Mellon University for their valuable comments and discussions.

References

[1] Arman, F. and Aggarwal, J. K. *Automatic generation of recognition strategies using CAD models.* **Proc... IEEE Workshop on Directions in Automated CAD-Based Vision,** pp. 124-133 (1991).

[2] Balakumar, P., Robert, J. C., Hoffman, R., Ikeuchi, K., and Kanade, T. *Vantage: A Frame-Based Geometric/Sensor Modeling System - Programmer/User's Manual v1.0,* **CMU-RI-TR-91-31,** The Robotics Institute e, Carnegie Mellon University (1991).

[3] Betz, J. W., Prince, J. L., and Bello, M. G. *Representation and transformation of uncertainty in evidence theroy framework.* **Proc. IEEE Conf. on Computer Vision and Pattern Recognition,** pp. 646-652 (1989).

[4] Bolles, R. C. and Horaud, P. *3DPO: A three-dimensional part orientation system.* In T. Kanade, editor, **Three-Dimensional Machine Vision,** pp. 399-450. Kluwer, Boston, MA (1987).

[5] Camps, O. I., Shapiro, L. G., and Haralick, R. M. *PREMIO: an overview.* **Proc. IEEE Workshop on Directions in Automated CAD-Based Vision,** pp. 11-21 (1991).

[6] Chen, S. and Freeman, H. *On the characteristic views of quadric-surfaced solids.* **Proc. IEEE Workshop on Directions in Automated CAD-Based Vision,** pp. 34-43 (1991).

[7] Flynn, P. J. and Jain, A. K. *BONSAI: 3D object recognition using constrained search.* **IEEE Trans. on Pattern Analysis and Machine Intelligence,** 13(10):1066-1075 (1991).

[8] Fujiwara, Y., Nayar, S., and Ikeuchi, K. *Appearance Simulator for Computer Vision Research.* **CMU-RI-TR-91-16,** The Robotics Institute, Carnegie Mellon University (1991).

[9] Goad, C. *Special purpose automatic programming for 3D model-based vision.* **Proc. Image Understanding Workshop,** pp. 94-104 (1983).

[10] Hansen, C. and Henderson, T. C. *CAGD-based computer vision.* **IEEE Trans. on Pattern Analysis and Machine Intelligence,** 11(11):1181-1193 (1989).

[11] Hebert, M. and Kanade, T. *Outdoor scene analysis using range data.* **Proc. Int. Conference on Robotics and Automation,** pp. 1426-1432 (1986).

[12] Hong, K. S., Ikeuchi, K., and Gremban, K. D. *Minimum cost aspect classification: A module of a vision algorithm compiler.* **Proc. 10th Int. Conf. on Pattern Recognition,** pp. 65-69 (1990). A longer version is available as **CMU-CS-90-124,** School of Computer Science, Carnegie Mellon University (1990).

[13] Ikeuchi, K. *Determining a depth map using a dual photometric stereo.* **Int J. Robotics Research,** 6(1):15-31 (1987).

[14] Ikeuchi, K. *Generating an interpretation tree from a CAD model for 3D-object recognition in bin-picking tasks.* **Int. J. Computer Vision,** 1(2):145-165 (1987).

[15] Ikeuchi, K. and Hong, K. S. *Determining linear shape change: toward automatic generation of object recognition programs.* **CVGIP: Image Understanding,** 53(2):154-170 (1991).

[16] Ikeuchi, K. and Kanade, T. *Automatic generation of object recognition programs.* **Proc. of the IEEE,** 76(8):1016-1035 (1988).

[17] Koenderink, J. J. and van Doorn, A. J. *The internal representation of solid shape with respect to vision.*

Biological Cybernetics, 32: 211-216 (1979).

[18] Kriegman, D. J. and Ponce, J. *Computing exact aspect graphs of curved objects: solids of revolution.* **Int. Journal of Computer Vision,** 5(2):119-135 (1990).

[19] Plantinga, H. and Dyer, C. R. *Visibility, occlusion, and the aspect graph.* **Int. Journal of Computer Vision,** 5(2):137-160 (1990).

[20] Sato, K., Ikeuchi, K., and Kanade, T. *Model based recognition of specular objects using sensor models.* **CVGIP: Image Understanding,** 55(2):155-169 (1992).

[21] Shafer, G. **A Mathematical Theory of Evidence.** Princeton Univ. Press, Princeton, NJ (1976).

[22] Suetens, P., Fua, P., and Hanson, A. J. *Computational strategies for object recognition.* **ACM Computing Surveys,** 24(1):5-61 (1992).